Decision Tree Learning

[read Chapter 3]
[recommended exercises 3.1, 3.4]

- Decision tree representation
- ID3 learning algorithm
- Entropy, Information gain
- Overfitting

A Tree to Predict C-Section Risk

Learned from medical records of 1000 women
Negative examples are C-sections

\[ [833+, 167-] .83+ .17- \]
Fetal_Presentation = 1: [822+, 116-] .88+ .12-  
| Previous_Csection = 0: [767+, 81-] .90+ .10-  
| | Primiparous = 0: [399+, 13-] .97+ .03-  
| | Primiparous = 1: [368+, 68-] .84+ .16-  
| | | Fetal_Distress = 0: [334+, 47-] .88+ .12-  
| | | | Birth_Weight < 3349: [201+, 10.6-] .95+ .05-  
| | | | Birth_Weight >= 3349: [133+, 36.4-] .78+ .22-  
| | | | Fetal_Distress = 1: [34+, 21-] .62+ .38-  
| | | | Previous_Csection = 1: [55+, 35-] .61+ .39-  
Fetal_Presentation = 2: [3+, 29-] .11+ .89-  
Fetal_Presentation = 3: [8+, 22-] .27+ .73-

Decision Tree for *PlayTennis*

Decision Trees

Decision tree representation:
- Each internal node tests an attribute
- Each branch corresponds to attribute value
- Each leaf node assigns a classification

How would we represent:
- \( \land, \lor, \text{XOR} \)
- \( (A \land B) \lor (C \land \neg D \land E) \)
- \( M \text{ of } N \)
When to Consider Decision Trees

- Instances describable by attribute–value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data

Examples:
- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences

Top-Down Induction of Decision Trees

Main loop:
1. \( A \leftarrow \) the “best” decision attribute for next node
2. Assign \( A \) as decision attribute for node
3. For each value of \( A \), create new descendant of node
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Which attribute is best?

![Decision Tree Diagram]

Entropy

\[ Entropy(S) = \text{expected number of bits needed to encode class } \oplus \text{ or } \ominus \text{ of randomly drawn member of } S \text{ (under the optimal, shortest-length code)} \]

Why?

Information theory: optimal length code assigns \(-\log_2 p\) bits to message having probability \(p\).

So, expected number of bits to encode \(\oplus\) or \(\ominus\) of random member of \(S\):

\[ p_\oplus(-\log_2 p_\oplus) + p_\ominus(-\log_2 p_\ominus) \]

\[ Entropy(S) = -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus \]
Information Gain

$Gain(S, A) = \text{expected reduction in entropy due to sorting on } A$

$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$

![Decision Tree Diagram]

Training Examples

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

Selecting the Next Attribute

Which attribute is the best classifier?

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Humidity</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>Normal</td>
</tr>
<tr>
<td>Wind</td>
<td>Strong</td>
</tr>
<tr>
<td></td>
<td>Weak</td>
</tr>
</tbody>
</table>

Gain (S, Humidity) = 0.965
Gain (S, Wind) = 0.940

Gain (S, Temperature) = 0.970
Gain (S, Wind) = 0.918

Which attribute should be tested here?

$S_{sunny} = \{D1, D2, ..., D14\}$

Gain ($S_{sunny}$, Humidity) = 0.970
Gain ($S_{sunny}$, Temperature) = 0.970
Gain ($S_{sunny}$, Wind) = 0.918
**Hypothesis Space Search by ID3**

- Hypothesis space is complete!
  - Target function surely in there...
- Outputs a single hypothesis (which one?)
  - Can’t play 20 questions...
- No back tracking
  - Local minima...
- Statistically-based search choices
  - Robust to noisy data...
- Inductive bias: approx “prefer shortest tree”

---

**Inductive Bias in ID3**

Note $H$ is the power set of instances $X$

→ Unbiased?

Not really...

- Preference for short trees, and for those with high information gain attributes near the root
- Bias is a *preference* for some hypotheses, rather than a *restriction* of hypothesis space $H$
- Occam’s razor: prefer the shortest hypothesis that fits the data

---

**Occam’s Razor**

Why prefer short hypotheses?

**Argument in favor:**

- Fewer short hyps. than long hyps.
  → a short hyp that fits data unlikely to be coincidence
- A long hyp that fits data might be coincidence

**Argument opposed:**

- There are many ways to define small sets of hyps
  - e.g., all trees with a prime number of nodes that use attributes beginning with “Z”
- What’s so special about small sets based on *size* of hypothesis??
Overfitting in Decision Trees

Consider adding noisy training example #15:
Sunny, Hot, Normal, Strong, PlayTennis = No
What effect on earlier tree?

Overfitting

Consider error of hypothesis \( h \) over
- training data: \( \text{error}_{\text{train}}(h) \)
- entire distribution \( D \) of data: \( \text{error}_D(h) \)

Hypothesis \( h \in H \) overfits training data if there is an alternative hypothesis \( h' \in H \) such that
\[
\text{error}_{\text{train}}(h) < \text{error}_{\text{train}}(h')
\]
and
\[
\text{error}_D(h) > \text{error}_D(h')
\]

Overfitting in Decision Tree Learning

![Graph showing accuracy vs. size of tree]

Overfitting

Avoiding Overfitting

How can we avoid overfitting?
- stop growing when data split not statistically significant
- grow full tree, then post-prune

How to select “best” tree:
- Measure performance over training data
- Measure performance over separate validation data set
- \( \text{MDL: minimize } size(tree) + size(misclassifications(tree)) \)
**Reduced-Error Pruning**

Split data into *training* and *validation* set

Do until further pruning is harmful:
1. Evaluate impact on *validation* set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves *validation* set accuracy
   - produces smallest version of most accurate subtree
   - What if data is limited?

---

**Rule Post-Pruning**

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others
3. Sort final rules into desired sequence for use

Perhaps most frequently used method (e.g., C4.5)

---

**Converting A Tree to Rules**

```
Outlook
  Sunny
  Overcast
Rain
  Yes
Humidity
  High
  Normal
  Strong
  Weak
Wind
  No
  Yes
```
Continuous Valued Attributes

Create a discrete attribute to test continuous
- \( \text{Temperature} = 82.5 \)
- \( (\text{Temperature} > 72.3) = t, f \)

<table>
<thead>
<tr>
<th>Temperature</th>
<th>40</th>
<th>48</th>
<th>60</th>
<th>72</th>
<th>80</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlayTennis</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Attributes with Many Values

Problem:
- If attribute has many values, Gain will select it
- Imagine using Date = June 3, 1996 as attribute

One approach: use GainRatio instead

\[
\text{GainRatio}(S, A) \equiv \frac{\text{Gain}(S, A)}{\text{SplitInformation}(S, A)}
\]

\[
\text{SplitInformation}(S, A) \equiv - \sum_{i=1}^{V} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}
\]

where \( S_i \) is subset of \( S \) for which \( A \) has value \( v_i \)

Attributes with Costs

Consider
- medical diagnosis, BloodTest has cost $150
- robotics, Width from Ceiling has cost 23 sec.

How to learn a consistent tree with low expected cost?

One approach: replace gain by

- Tan and Schlimmer (1990)

\[
\frac{\text{Gain}^2(S, A)}{\text{Cost}(A)}
\]

- Nunez (1988)

\[
\frac{2 \text{Gain}(S, A) - 1}{(\text{Cost}(A) + 1)^w}
\]

where \( w \in [0, 1] \) determines importance of cost
Unknown Attribute Values

What if some examples missing values of $A$?
Use training example anyway, sort through tree

- If node $n$ tests $A$, assign most common value of $A$ among other examples sorted to node $n$
- Assign most common value of $A$ among other examples with same target value
- Assign probability $p_i$ to each possible value $v_i$ of $A$
  - Assign fraction $p_i$ of example to each descendant in tree

Classify new examples in same fashion